Somatic Variant Detection Workflow

Genomics and Molecular Pathology Lab

UT Southwestern Medical Center

Author: Brandi Cantarel

Version: 1.0

Effective Date: 06/08/2017

1. Introduction

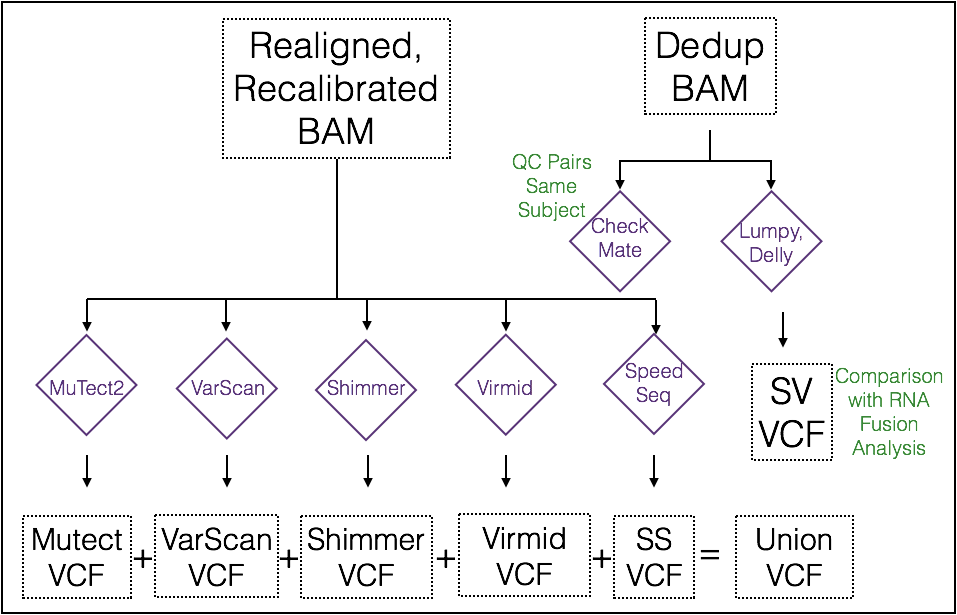
This SOP describes the analysis pipeline of somatic variant calling. This pipeline starts with the input of alignment files of normal and tumor samples and identifies somatic variants. It provides better sensitivity by integrating variant calls from various programs including SpeedSeq, MuTect, VarScan, Shimmer, and Virmid.

2. Requirements

2.1. Data requirements

The pipeline requires as input BAM files generated by the alignment step of genome/exome/target panel sequencing data analysis. The pipeline also requires the following reference data: (1) the FASTA sequence of a reference genome, currently Human GRCh38; (2) the target enrichment regions; (3) a set of reference single nucleotide polymorphisms (SNPs) from the NCBI SNP database; (4) a set of somatic mutations from COSMIC database and (5) a set of reference indels from the 1000 genomes project.

3. Procedures

Figure 1: Procedure Overview

3.1. Index BAM files

|  |  |
| --- | --- |
| executable | speedseq/20160506 samtools/intel/1.3 |
| input | tumor BAM file, normal BAM file |
| output | tumor BAM index file (bai), normal BAM index file (bai) |
| command | >sambamba index -t \$SLURM\_CPUS\_ON\_NODE ${tumor}  >sambamba index -t \$SLURM\_CPUS\_ON\_NODE ${normal} |

3.2. Checkmates

Determine the similarity between sample pairs to ensure samples are from the same subject

|  |  |
| --- | --- |
| executable | python/2.7.x-anaconda |
| input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions |
| output | Text file with similarity score between samples |
| command | >python /project/shared/bicf\_workflow\_ref/seqprg/NGSCheckMate/ncm.py -B -d ./ -bed ${index\_path}/NGSCheckMate.bed -O ./ -N ${tid}\_${nid} |

3.3. Structural Variant Calling

Identify structural variants.

|  |  |
| --- | --- |
| executable | bcftools/intel/1.3 samtools/intel/1.3 bedtools/2.25.0 speedseq/20160506 snpeff/4.2 vcftools/0.1.14 novoBreak\_distribution\_v1.1.3rc |
| input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions |
| output | somatic variant calls in VCF |
| command | >perl $baseDir/scripts/make\_delly\_sample.pl ${tid} ${nid}  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly call -t TRA -o delly\_translocations.bcf -q 30 -g ${reffa} ${tumor} ${normal}  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly call -t DUP -o delly\_duplications.bcf -q 30 -g ${reffa} ${tumor} ${normal}  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly call -t INV -o delly\_inversions.bcf -q 30 -g ${reffa} ${tumor} ${normal}  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly call -t DEL -o delly\_deletion.bcf -q 30 -g ${reffa} ${tumor} ${normal}  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly call -t INS -o delly\_insertion.bcf -q 30 -g ${reffa} ${tumor} ${normal}  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly filter -t TRA -o delly\_tra.bcf -f somatic -s samples.tsv delly\_translocations.bcf  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly filter -t DUP -o delly\_dup.bcf -f somatic -s samples.tsv delly\_translocations.bcf  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly filter -t INV -o delly\_inv.bcf -f somatic -s samples.tsv delly\_translocations.bcf  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly filter -t DEL -o delly\_del.bcf -f somatic -s samples.tsv delly\_translocations.bcf  >/project/BICF/BICF\_Core/shared/seqprg/delly/src/delly filter -t INS -o delly\_ins.bcf -f somatic -s samples.tsv delly\_translocations.bcf  >bcftools concat -a -O v delly\_dup.bcf delly\_inv.bcf delly\_tra.bcf delly\_del.bcf delly\_ins.bcf| vcf-sort -t temp > ${tid}\_${nid}.delly.vcf  >perl $baseDir/scripts/vcf2bed.sv.pl ${tid}\_${nid}.delly.vcf > delly.bed  >bgzip ${tid}\_${nid}.delly.vcf  >tabix ${tid}\_${nid}.delly.vcf.gz  >bcftools view -O z -o delly.vcf.gz -s ${tid} ${tid}\_${nid}.delly.vcf.gz  >/project/BICF/BICF\_Core/shared/seqprg/novoBreak\_distribution\_v1.1.3rc/run\_novoBreak.sh /project/BICF/BICF\_Core/shared/seqprg/novoBreak\_distribution\_v1.1.3rc ${reffa} ${tumor} ${normal} \$SLURM\_CPUS\_ON\_NODE  >perl $baseDir/scripts/vcf2bed.sv.pl novoBreak.pass.flt.vcf |sort -T temp -V -k 1,1 -k 2,2n > novobreak.bed  >mv novoBreak.pass.flt.vcf ${tid}\_${nid}.novobreak.vcf  >bgzip ${tid}\_${nid}.novobreak.vcf  >sambamba sort -t \$SLURM\_CPUS\_ON\_NODE -n -o tumor.namesort.bam ${tumor}  >sambamba sort -t \$SLURM\_CPUS\_ON\_NODE -n -o normal.namesort.bam ${normal}  >sambamba view -h tumor.namesort.bam | samblaster -M -a --excludeDups --addMateTags --maxSplitCount 2 --minNonOverlap 20 -d discordants.sam -s splitters.sam > temp.sam  >gawk '{ if (\$0~"^@") { print; next } else { \$10="\*"; \$11="\*"; print } }' OFS="\\t" splitters.sam | samtools view -S -b - | samtools sort -o tumor.splitters.bam -  >gawk '{ if (\$0~"^@") { print; next } else { \$10="\*"; \$11="\*"; print } }' OFS="\\t" discordants.sam | samtools view -S -b - | samtools sort -o tumor.discordants.bam -  >sambamba view -h normal.namesort.bam | samblaster -M -a --excludeDups --addMateTags --maxSplitCount 2 --minNonOverlap 20 -d discordants.sam -s splitters.sam > temp.sam  >gawk '{ if (\$0~"^@") { print; next } else { \$10="\*"; \$11="\*"; print } }' OFS="\\t" splitters.sam | samtools view -S -b - | samtools sort -o normal.splitters.bam -  >gawk '{ if (\$0~"^@") { print; next } else { \$10="\*"; \$11="\*"; print } }' OFS="\\t" discordants.sam | samtools view -S -b - | samtools sort -o normal.discordants.bam -  >speedseq sv -t \$SLURM\_CPUS\_ON\_NODE -o ${tid}\_${nid}.sssv -R ${reffa} -B ${normal},${tumor} -D normal.discordants.bam,tumor.discordants.bam -S normal.splitters.bam,tumor.splitters.bam -x ${index\_path}/exclude\_alt.bed  java -jar \$SNPEFF\_HOME/SnpSift.jar filter "GEN[0].SU < 1 & GEN[1].SU > 2" ${tid}\_${nid}.sssv.sv.vcf.gz > lumpy.vcf  >perl $baseDir/scripts/vcf2bed.sv.pl lumpy.vcf > lumpy.bed  >bcftools view -O z -o sssv.vcf.gz -s ${tid} ${tid}\_${nid}.sssv.sv.vcf.gz  >bedtools multiinter -cluster -header -names novobreak delly lumpy -i novobreak.bed delly.bed lumpy.bed > sv.intersect.bed  >grep novobreak sv.intersect.bed |cut -f 1,2,3 |sort -V -k 1,1 -k 2,2n |grep -v start | bedtools intersect -header -b stdin -a ${tid}\_${nid}.novobreak.vcf.gz | perl -p -e 's/SPIKEIN/${tid}/' |bgzip > t1.vcf.gz  >grep delly sv.intersect.bed |cut -f 1,2,3 |sort -V -k 1,1 -k 2,2n |grep -v 'start' |grep -v 'novobreak' | bedtools intersect -header -b stdin -a delly.vcf.gz |bgzip > t2.vcf.gz  >grep lumpy sv.intersect.bed |cut -f 1,2,3 |sort -V -k 1,1 -k 2,2n |grep -v 'start' |grep -v 'delly' |grep -v 'novobreak' | bedtools intersect -header -b stdin -a sssv.vcf.gz |bgzip > t3.vcf.gz  >vcf-concat t1.vcf.gz t2.vcf.gz t3.vcf.gz |vcf-sort -t temp > ${tid}\_${nid}.sv.vcf  >perl $baseDir/scripts/vcf2bed.sv.pl ${tid}\_${nid}.sv.vcf |sort -V -k 1,1 -k 2,2n | grep -v 'alt' |grep -v 'random' |uniq > svs.bed  >bedtools intersect -header -wb -a svs.bed -b ${index\_path}/gencode.exons.bed > exonoverlap\_sv.txt  >bedtools intersect -v -header -wb -a svs.bed -b ${index\_path}/gencode.exons.bed | >bedtools intersect -header -wb -a stdin -b ${index\_path}/gencode.genes.chr.bed > geneoverlap\_sv.txt  >perl $baseDir/scripts/annot\_sv.pl -r ${index\_path} -i ${tid}\_${nid}.sv.vcf  >bgzip ${tid}\_${nid}.sv.vcf |

3.4. Somatic variant calling using SpeedSeq

Detect somatic variants in target regions using SpeedSeq. The variants are filtered based on the quality score (>= 10) and read depth (>= 10).

|  |  |
| --- | --- |
| executable | python/2.7.x-anaconda bedtools/2.25.0 snpeff/4.2 speedseq/20160506 bcftools/intel/1.3 vcftools/0.1.14 |
| input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions |
| output | somatic variant calls in VCF |
| command | >speedseq somatic -q 10 -w ${target\_panel} -t 30 -o ${tid}.sssom ${reffa} ${normal} ${tumor}  >vcf-annotate -H -n --fill-type ${tid}.sssom.vcf.gz | java -jar SnpSift.jar filter --pass '((QUAL >= 10) & (GEN[\*].DP >= 10))' | perl -pe 's/TUMOR/${tid}/' | perl -pe 's/NORMAL/${nid}/g' | bgzip > ${tid}\_${nid}.sspanel.vcf.gz |

3.3. Somatic variant calling using MuTect

Detect somatic variants with confidence quality >= 10 using MuTect. The variants are filtered based on the fisher strand value (<= 60) and read depth (>= 10).

|  |  |
| --- | --- |
| executable | python/2.7.x-anaconda gatk/3.5 bcftools/intel/1.3 bedtools/2.25.0 snpeff/4.2 vcftools/0.1.14 |
| input | tumor BAM file, normal BAM file, reference genome FASTA file, dbSNP VCF file, COSMIC VCF file |
| output | somatic variant calls in VCF |
| command | >cut -f 1 ${index\_path}/genomefile.chr.txt | xargs -I {} -n 1 -P 10 sh -c "java -Xmx10g -jar GenomeAnalysisTK.jar -R ${reffa} -D ${dbsnp} -T MuTect2 -stand\_call\_conf 30 -stand\_emit\_conf 10.0 -A FisherStrand -A QualByDepth -A VariantType -A DepthPerAlleleBySample -A HaplotypeScore -A AlleleBalance -I:tumor ${tumor} -I:normal ${normal} --cosmic ${cosmic} -o ${tid}.{}.mutect.vcf -L {}"  >vcf-concat ${tid}\*.vcf | vcf-sort | vcf-annotate -n --fill-type | java -jar SnpSift.jar filter -p '((FS <= 60) & GEN[\*].DP >= 10)' | perl -pe 's/TUMOR/${tid}/' | perl -pe 's/NORMAL/${nid}/g' | bgzip > ${tid}\_${nid}.pmutect.vcf.gz |

3.4. Somatic variant calling using VarScan

Detect somatic and the other types of variants in target regions supported by reads with mapping quality >= 50 using VarScan. The somatic variants are filtered based on the read depth (>= 10).

|  |  |
| --- | --- |
| executable | python/2.7.x-anaconda bedtools/2.25.0 snpeff/4.2 bcftools/intel/1.3 samtools/intel/1.3 VarScan/2.4.2 speedseq/20160506 vcftools/0.1.14 |
| input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions |
| output | somatic variant calls in VCF |
| command | >sambamba mpileup -L ${target\_panel} -t 30 ${tumor} --samtools "-C 50 -f ${reffa}" > t.mpileup  >sambamba mpileup -L ${target\_panel} -t 30 ${normal} --samtools "-C 50 -f ${reffa}" > n.mpileup  >VarScan somatic n.mpileup t.mpileup ${tid}.vscan --output-vcf 1  >VarScan copynumber n.mpileup t.mpileup ${tid}.vscancnv  >vcf-concat ${tid}.vscan\*.vcf | vcf-sort | vcf-annotate -n --fill-type -n | java -jar SnpSift.jar filter '((exists SOMATIC) & (GEN[\*].DP >= 10))' | perl -pe 's/TUMOR/${tid}/' | perl -pe 's/NORMAL/${nid}/g' | bedtools intersect -header -a stdin -b ${target\_panel} | bgzip > ${tid}\_${nid}.varscan.vcf.gz |

3.5. Somatic variant calling using Shimmer

Detect somatic variants with the quality score >= 25 using Shimmer. The variants are filtered based on the target regions and read depth (>= 10).

|  |  |
| --- | --- |
| executable | python/2.7.x-anaconda bedtools/2.25.0 snpeff/4.2 bcftools/intel/1.3 shimmer/0.1.1 vcftools/0.1.14 |
| input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions |
| output | somatic variant calls in VCF |
| command | >shimmer.pl --minqual 25 --ref ${reffa} ${normal} ${tumor} --outdir shimmer 2> shimmer.err  >perl add\_readct\_shimmer.pl  >vcf-annotate -n --fill-type shimmer/somatic\_diffs.readct.vcf | java -jar SnpSift.jar filter '(GEN[\*].DP >= 10)' | perl -pe 's/TUMOR/${tid}/' | perl -pe 's/NORMAL/${nid}/g' | bedtools intersect -header -a stdin -b ${target\_panel} | bgzip > ${tid}\_${nid}.shimmer.vcf.gz |

3.6. Somatic variant calling using Virmid

Detect somatic variants using Virmid. The variants are filtered based on the target regions and read depth (>= 10).

|  |  |
| --- | --- |
| executable | python/2.7.x-anaconda bedtools/2.25.0 snpeff/4.2 virmid/1.2 vcftools/0.1.14 |
| input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions |
| output | somatic variant calls in VCF |
| command | >virmid -R ${reffa} -D ${tumor} -N ${normal} -s $cosmic t 30 -M 2000 -c1 10 -c2 10  >perl addgt\_virmid.pl ${tumor}.virmid.som.passed.vcf  >perl addgt\_virmid.pl ${tumor}.virmid.loh.passed.vcf  >vcf-concat \*gt.vcf | vcf-sort | vcf-annotate -n --fill-type -n | java -jar SnpSift.jar filter '((NDP >= 10) & (DDP >= 10))' | perl -pe 's/TUMOR/${tid}/' | perl -pe 's/NORMAL/${nid}/g' | bedtools intersect -header -a stdin -b ${target\_panel} |bgzip > ${tid}\_${nid}.virmid.vcf.gz |

3.7. Generate union

Integrate result somatic variant lists generated by SpeedSeq, Mutect, VarScan, Shimmer, and Virmid.

|  |  |
| --- | --- |
| executable | gatk/3.5 python/2.7.x-anaconda bedtools/2.25.0 snpeff/4.2 bcftools/intel/1.3 samtools/intel/1.3 |
| input | VCF files generated by SpeedSeq, MuTect, VarScan, Shimmer, and Virmid |
| output | union variant calls in VCF |
| command | >perl somatic\_unionize\_vcf.pl -r ${index\_path} ${ss} ${mutect} ${shimmer} ${vscan} ${virmid}  >sh integrate.sh  >perl somatic\_uniform\_integrated\_vcf.pl ${fname}.temp.vcf  >bgzip ${fname}.union.vcf |

3.8. Annotate VCF

Annotate VCF with publicly available genes

|  |  |
| --- | --- |
| executable | python/2.7.x-anaconda bedtools/2.25.0 snpeff/4.2 bcftools/intel/1.3 samtools/intel/1.3 |
| input | reference genome fasta, union vcf |
| output | annotated vcf |
| command | >bcftools annotate -Oz -a ${index\_path}/ExAC.vcf.gz -o ${fname}.exac.vcf.gz --columns CHROM,POS,AC\_Het,AC\_Hom,AC\_Hemi,AC\_Adj,AN\_Adj,AC\_POPMAX,AN\_POPMAX,POPMAX ${unionvcf}  >tabix ${fname}.exac.vcf.gz  >bcftools annotate -Oz -a ${index\_path}/dbSnp.vcf.gz -o ${fname}.dbsnp.vcf.gz --columns CHROM,POS,ID,RS ${fname}.exac.vcf.gz  >tabix ${fname}.dbsnp.vcf.gz  >bcftools annotate -Oz -a ${index\_path}/clinvar.vcf.gz -o ${fname}.clinvar.vcf.gz --columns CHROM,POS,CLNSIG,CLNDSDB,CLNDSDBID,CLNDBN,CLNREVSTAT,CLNACC ${fname}.dbsnp.vcf.gz  >tabix ${fname}.clinvar.vcf.gz  >bcftools annotate -Oz -a ${index\_path}/utswv2\_artifact.bed.gz -o ${fname}.utswbl.vcf.gz -m "UTSWBlacklist" -c CHROM,FROM,TO ${fname}.clinvar.vcf.gz  >tabix ${fname}.utswbl.vcf.gz  >java -Xmx10g -jar \$SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c \$SNPEFF\_HOME/snpEff.config ${snpeff\_vers} ${fname}.utswbl.vcf.gz | java -jar \$SNPEFF\_HOME/SnpSift.jar annotate ${index\_path}/cosmic.vcf.gz - | java -Xmx10g -jar \$SNPEFF\_HOME/SnpSift.jar dbnsfp -v -db ${index\_path}/dbNSFP.txt.gz - | java -Xmx10g -jar \$SNPEFF\_HOME/SnpSift.jar gwasCat -db ${index\_path}/gwas\_catalog.tsv - |bgzip > ${fname}.annot.vcf.gz  >tabix ${fname}.annot.vcf.gz  >bcftools stats ${fname}.annot.vcf.gz > ${fname}.stats.txt  >plot-vcfstats -s -p ${fname}.statplot ${fname}.stats.txt |

4. Implementation

The Workflow used in this SOP can be downloaded here: <https://git.biohpc.swmed.edu/brandi.cantarel/clinseq_workflows>

Usage:

nextflow run somatic.nf –input input\_directory –output output\_directory –design samplesheet.txt

where the input directory is the name of the folder where the BAM files reside, the output directory is the name of the folder where the output files will be written and the samplesheet.txt is a tab delimited file that contains the sample id, subject id and BAM files.

5. Related Documents & References

* SAMtools: <http://samtools.sourceforge.net/>
  + Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R; 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009 Aug 15;25(16):2078-9.
  + Li H. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. Bioinformatics. 2011 Nov 1;27(21):2987-93.
* Sambamba: <http://lomereiter.github.io/sambamba/>
  + Tarasov A, Vilella AJ, Cuppen E, Nijman IJ, Prins P. Sambamba: fast processing of NGS alignment formats. Bioinformatics. 2015 Jun 15;31(12):2032-4.
* SpeedSeq: <https://github.com/hall-lab/speedseq>
  + Chiang C, Layer RM, Faust GG, Lindberg MR, Rose DB, Garrison EP, Marth GT, Quinlan AR, Hall IM. SpeedSeq: ultra-fast personal genome analysis and interpretation. Nat Methods. 2015 Oct;12(10):966-8.
* MuTect: <https://software.broadinstitute.org/gatk/documentation/tooldocs/current/org_broadinstitute_gatk_tools_walkers_cancer_m2_MuTect2.php>
  + <http://archive.broadinstitute.org/cancer/cga/mutect>
* COSMIC: <http://cancer.sanger.ac.uk/cosmic>
* VarScan: <http://dkoboldt.github.io/varscan/>
  + Koboldt DC, Zhang Q, Larson DE, Shen D, McLellan MD, Lin L, Miller CA, Mardis ER, Ding L, Wilson RK. VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. Genome Res. 2012 Mar;22(3):568-76.
* Shimmer: <https://github.com/nhansen/Shimmer>
  + Hansen NF, Gartner JJ, Mei L, Samuels Y, Mullikin JC. Shimmer: detection of genetic alterations in tumors using next-generation sequence data. Bioinformatics. 2013 Jun 15;29(12):1498-503.
* Virmid: <https://sourceforge.net/projects/virmid/>
  + Kim S, Jeong K, Bhutani K, Lee J, Patel A, Scott E, Nam H, Lee H, Gleeson JG, Bafna V. Virmid: accurate detection of somatic mutations with sample impurity inference. Genome Biol. 2013 Aug 29;14(8):R90.
* BEDTools: <http://bedtools.readthedocs.io/en/latest/>
  + Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010 Mar 15;26(6):841-2.
* VCFtools: <http://vcftools.sourceforge.net/index.html>
  + Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R; 1000 Genomes Project Analysis Group. The variant call format and VCFtools. Bioinformatics. 2011 Aug 1;27(15):2156-8.
* SnpSift: <http://snpeff.sourceforge.net/>
  + Cingolani P, Patel VM, Coon M, Nguyen T, Land SJ, Ruden DM, Lu X. Using Drosophila melanogaster as a Model for Genotoxic Chemical Mutational Studies with a New Program, SnpSift. Front Genet. 2012 Mar 15;3:35.

6. Revision History